

머신러닝 & 딥러닝 3

AI 학술동아리 <MLP>

- Index

- 1. 다중 분류**
- 2. 로지스틱 회귀**
- 3. 손실 함수**
- 4. 점진적 학습 - 경사 하강법**

1. Multiclass Classification(다중 분류)

- Binary Classification(이진 분류)

- 두 class 중 어떤 class인지 분류

k-NN Classification 그대로 사용 가능

- Multiclass Classification(다중 분류)

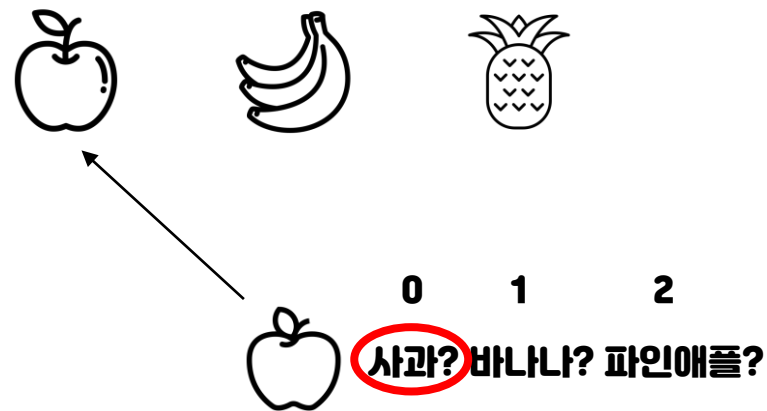
- 2개 이상의 class 중 어떤 class인지 분류

Binary Classification

각 class에 대한 확률이 나오고, 그 중 가장 큰 확률을 갖는 class로 예측



Multiclass Classification



Tip! scikit-learn에서는 문자열로 된 target값을 그대로 사용 가능

2. Logistic Regression(로지스틱 회귀)

이름은 Regression이지만, **Classification 모델**

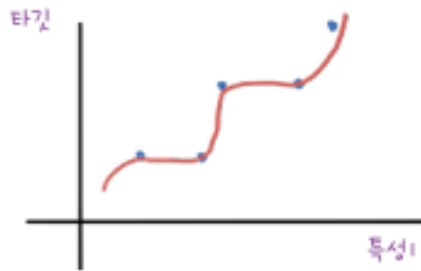
Linear Regression과 동일하게 선형 방정식을 학습 (z는 어떤 값도 가능)

예) $z = a * \text{feature1} + b * \text{feature2} + c * \text{feature3} + d * \text{feature4} + e$

C 매개변수 값으로 **regularization의 강도를 조절**(C값이 작을수록 규제 강도가 세짐)

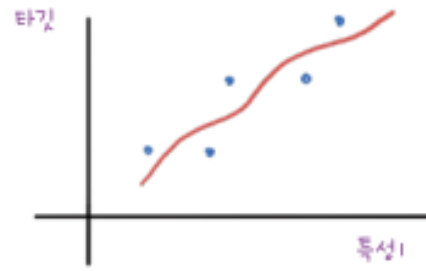
- C값이 작으면, 계수 값을 더 줄이고 조금 더 underfitting 되도록 만듦
- C값이 크면, 계수를 줄이는 역할이 줄어들고 overfitting 가능성이 큼

↓ 요약



C값이 클 때

→ 규제



C값이 작을 때

주의!
Ridge, Lasso Regression의 alpha와
Logistic Regression의 C는 반대

2. Logistic Regression(로지스틱 회귀) - Binary Classification

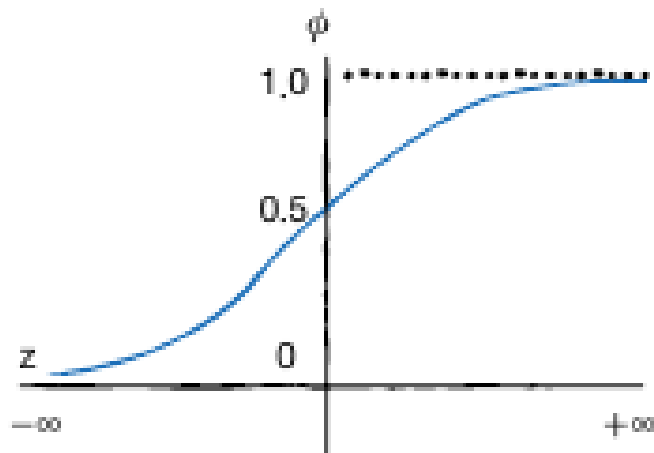
z는 어떤 값도 가능하지만, class에 대한 확률로 나타내기 위해서 **0~1사이 값이 되어야 함**
-> **sigmoid function**(시그모이드 함수)-(**logistic function**(로지스틱 함수))

sigmoid function : S자형 곡선 또는 시그모이드 곡선을 갖는 수학 함수.
- 대표적인 예 : **logistic function**

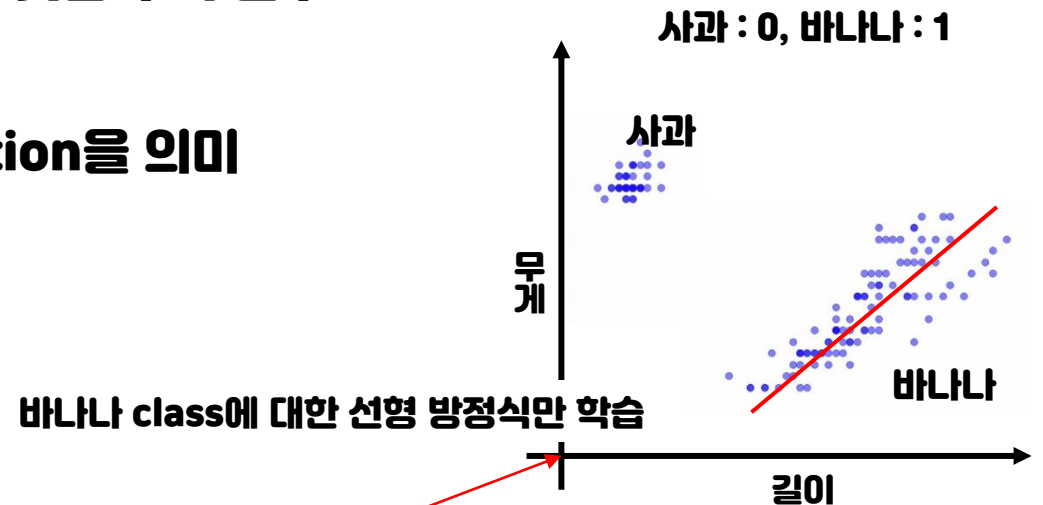
보통 ML에서 “**sigmoid function**”은 logistic function을 의미

$$\phi = \frac{1}{1 + e^{-z}}$$

시그모이드 함수



시그모이드 그래프



Binary Classification에서는 **선형 방정식을 한 개만 학습**

-> 양성 class에 대한 z값

-> **양성 class의 확률**

-> 음성 class 확률 = (1 - 양성 class 확률)

=> **두 class의 확률 중 큰 것을 해당 class로 판단**

2. Logistic Regression(로지스틱 회귀) - Multiclass Classification

z는 어떤 값도 가능하지만, class에 대한 확률로 나타내기 위해서 **0~1사이 값이 되어야 함**
그리고 판단해야 할 **class가 여러 개이기 때문에 z값도 class개수 만큼 있어야 함**
→ **softmax function**(소프트맥스 함수)

예)

$$e_sum = e^{z1} + e^{z2} + e^{z3} + e^{z4} + e^{z5} + e^{z6} + e^{z7}$$

$$s1 = \frac{e^{z1}}{e_sum}, s2 = \frac{e^{z2}}{e_sum}, \dots, s7 = \frac{e^{z7}}{e_sum}$$

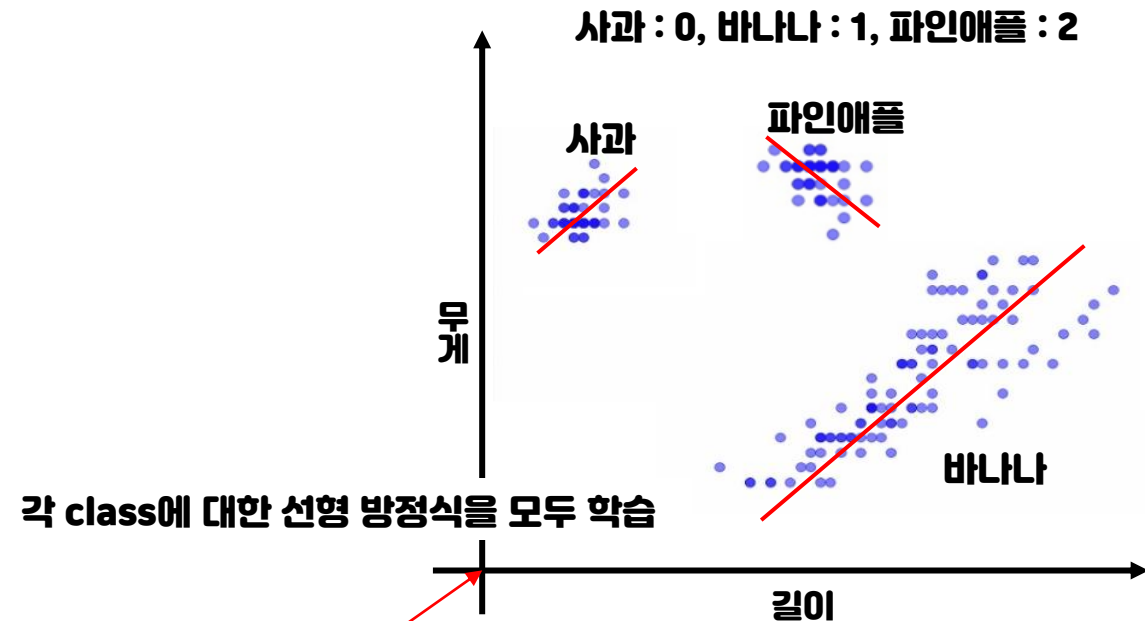
$$f(x) = \frac{1}{1 + e^{-x}}$$
$$\sigma(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{e^z + 1}$$
$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{i=1}^K e^{z_i}}$$

로지스틱 함수

sigmoid 시그모이드

↓ 일반화 generalization

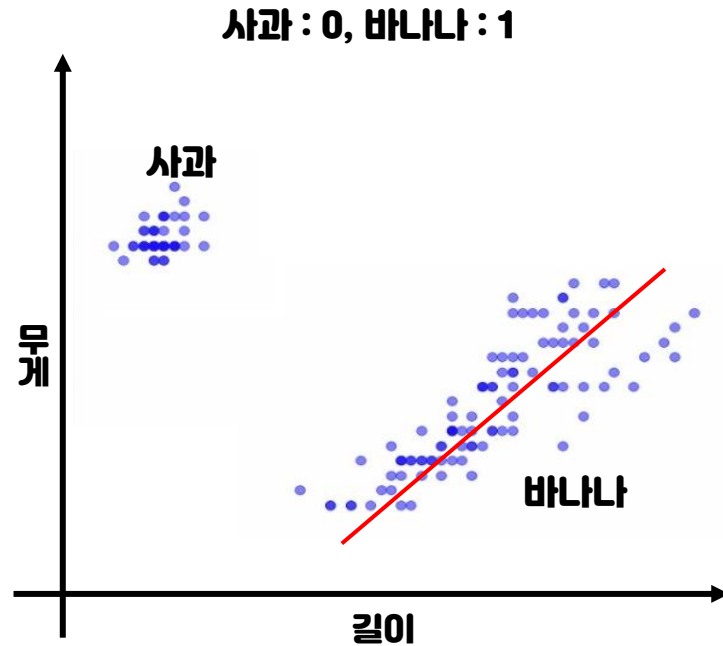
softmax 소프트맥스



Multiclass Classification에서는 **선형 방정식을 class개수 만큼 학습**
→ 각 class에 대한 z값
→ 각 class의 확률
⇒ **각 class의 확률 중 제일 큰 것을 해당 class로 판단**

2. Logistic Regression(로지스틱 회귀) 요약

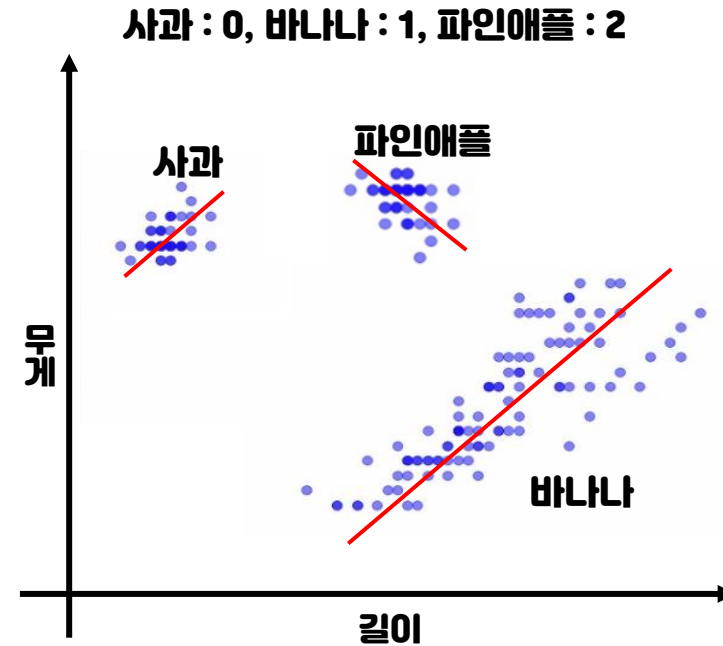
Binary Classification



바나나 class에 대한 선형 방정식만 학습

sigmoid function

Multiclass Classification



각 class에 대한 선형 방정식을 모두 학습

softmax function

3. Loss function(손실 함수)

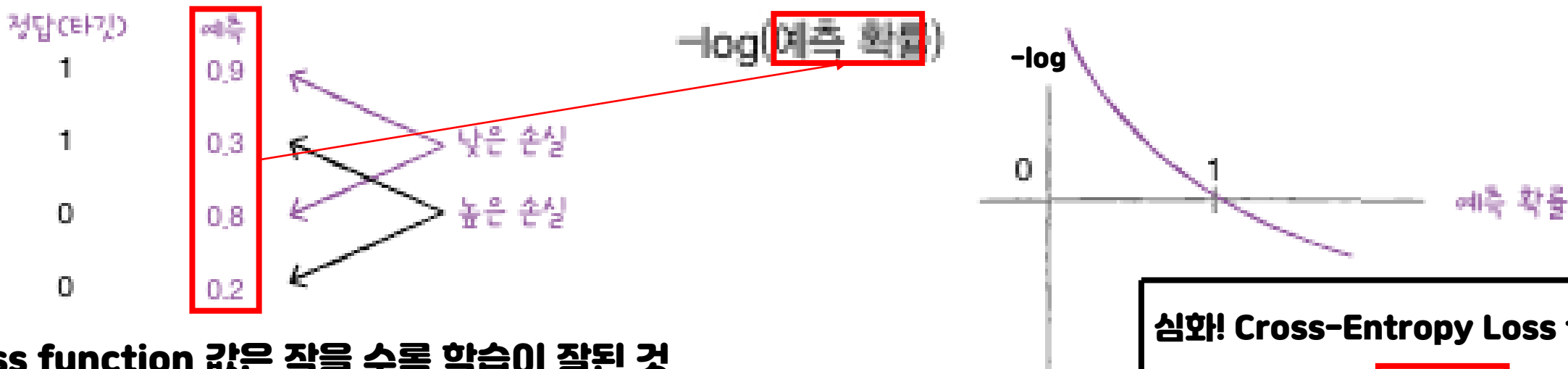
- **Loss function(손실 함수)** - 샘플 하나에 대한 손실
 - ML 알고리즘이 얼마나 엉터리인지 측정하는 기준(값이 작을 수록 학습이 잘 된 것)
- **Cost function(비용 함수)**
 - train set에 있는 모든 샘플에 대한 loss function의 합
 - 보통 **Loss function = Cost function**

Tip! Classification/Regression의 Loss function

- 대표적 예시 (다른 Loss function이 많이 있음)
- Classification
 - Binary classification
 - **Binary Cross-Entropy Loss function**(이진 크로스엔트로피 손실 함수)
= Logistic Loss function(로지스틱 손실 함수)
 - Multiclass classification
 - **Cross-Entropy Loss function**(크로스엔트로피 손실 함수)
- Regression
 - **Mean Squared Error**(평균 제곱 오차)

심화! Logistic Loss function

target class에 대한 예측 확률



Loss function 값은 작을 수록 학습이 잘된 것

예측 확률에 $-\log$ 를 취함

예측 확률이 0에 가까울수록(정답을 예측 잘 못할수록) Loss는 매우 큰 값
예측 확률이 1에 가까울수록(정답을 예측 잘 할수록) Loss는 매우 작은 값

심화! Cross-Entropy Loss function

target	predict	$-\log(\text{예측 확률})$
사과	0.6	Loss 값
파인애플	0.7	
바나나	0.9	
사과	0.4	
파인애플	0.2	

Tip! Multiclass Classification도 마찬가지로 target class에 대한 예측 확률에 $-\log$ 를 취해 Loss 값을 얻음

4. 점진적 학습

• 점진적 학습

- 앞서 **훈련한 모델을 버리지 않고, 새로운 데이터에 대해서만 조금씩 훈련**
 - > 훈련에 사용한 데이터를 모두 유지할 필요 X, 앞서 학습한 데이터를 잊어버릴 일 X
- 점진적 학습은 **Loss function 값을 점점 줄이는** 방향으로 학습하는 것

예) **Gradient Descent**(경사 하강법)

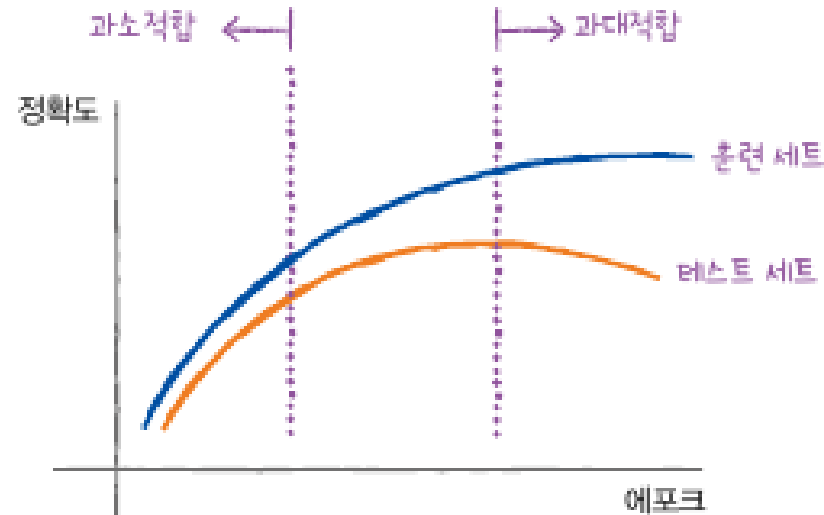
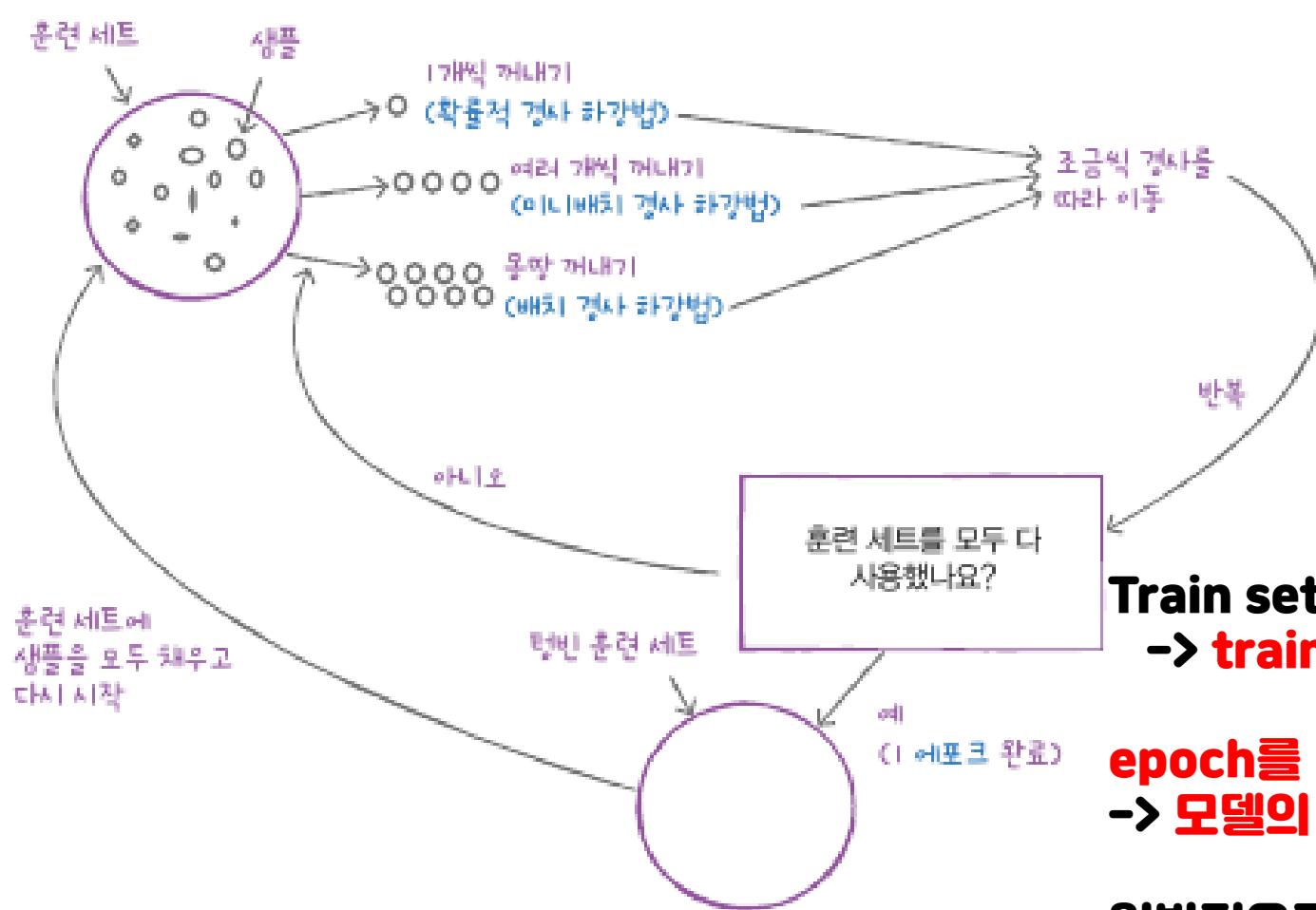
- Stochastic Gradient Descent(확률적 경사 하강법)
- Minibatch Gradient Descent(미니배치 경사 하강법)
- batch Gradient Descent(배치 경사 하강법)

4-1. Gradient Descent(경사 하강법)

- Stochastic Gradient Descent(확률적 경사 하강법)
 - Train set에서 **딱 하나의 샘플**을 **랜덤**으로 골라 Loss function 값을 줄이는 방향으로 학습하는 방법
- Minibatch Gradient Descent(미니배치 경사 하강법)
 - Train set에서 **몇 개의 샘플**을 **랜덤**으로 골라 Loss function 값을 줄이는 방향으로 학습하는 방법(**실제로 많이 사용하는 방법**)
- Batch Gradient Descent(배치 경사 하강법)
 - Train set에서 **전체 샘플**을 사용해서 Loss function 값을 줄이는 방향으로 학습하는 방법 (컴퓨터 자원을 너무 많이 사용)

Tip! **Neural Network 알고리즘**(Deep Learning)에서는 **stochastic gradient descent** 또는 **minibatch gradient descent**를 꼭 사용

4-1. Gradient Descent(경사 하강법)



Train set 전체 샘플을 사용할 때까지 반복
→ **train set를 모두 사용 = 1epoch(에포크)**


epoch를 반복 수행
→ **모델의 성능이 더 이상 좋아지지 않으면 종료**

일반적으로 Gradient Descent는 수십, 수백 번 이상 epoch 수행

Tip! Scikit-learn의 SGD


- **SGDClassifier**

- **모델 생성 할 때 사용**

- **loss 매개변수(기본값은 “hinge”)** : Loss function 설정
 - **penalty 매개변수(기본값은 “l2”)** : Regularization 종류 설정
 - **alpha 매개변수(기본값은 “0.0001”)** : Regularization 강도 설정
-  우리는 “**log_loss**”(logistic loss function) 사용

- **SGDRegressor**

- **모델 생성 할 때 사용**

- **loss 매개변수(기본값은 “squared_error”)** : Loss function 설정
 - **penalty 매개변수(기본값은 “l2”)** : Regularization 종류 설정
 - **alpha 매개변수(기본값은 “0.0001”)** : Regularization 강도 설정
-  = MSE